# Locality Sensitive Hashing (LSH) 101 for Atlas LSH Neural Networks

## 1. What is Locality Sensitive Hashing?

Conventional hash functions are designed so that even tiny changes in the input produce completely unrelated outputs. Locality Sensitive Hashing (LSH) does almost the opposite.

The central idea of LSH is:

> Similar inputs should, with reasonably high probability, produce similar hash codes.

In geometric terms, points that are close together in high-dimensional space tend to receive similar binary signatures, while distant points tend to receive different signatures.

For Atlas LSH neural networks, the LSH bits become a compact description of where an input lies within the geometry of the input space.

---

## 2. Random hyperplanes: slicing space

Suppose an input vector is

$x = (x_1, x_2, \dots , x_n)$

Generate a random vector

$r = (r_1, r_2, \dots ,r_n)$

and compute the dot product

$s = x \cdot r$

The sign of this projection determines a single LSH bit:

b = (x>0)? 1:0

Geometrically, the random vector defines a hyperplane passing through the origin.

The hyperplane divides space into two half-spaces:

- one side produces bit = 1,
- the other side produces bit = 0.

Each random hyperplane therefore asks a binary geometric question:

> "Which side of this hyperplane does the input lie on?"

With k independent random hyperplanes we obtain a k-bit code.

Each bit adds another partition of the space.

---

# 3. LSH as repeated geometric views of the data

A useful way to think about Atlas networks is:

> Every random projection is a different geometric view of the same input.

One projection may (weakly) distinguish cats from dogs.

Another may distinguish indoor scenes from outdoor scenes.

Another may distinguish texture.

Another may distinguish shape.

Individually, each projection is crude. Collectively, many projections provide a surprisingly informative description of where the input lies.

The collection of bits

$(b_1, b_2, \dots, b_k)$

acts like a geometric address.

Inputs that occupy nearby regions of space tend to share many bits.

---

# 4. Binarization as controlled information loss

The original projections are floating-point numbers:

$$s_i = x \cdot r_i$$

but Atlas systems often keep only the sign.

This discards magnitude information.

For example

$$+0.01 \rightarrow 1$$

and

$$+1000 \rightarrow 1$$

produce the same bit.

This is a deliberate information bottleneck.

The network intentionally throws away information in exchange for:

- compact representations,
- fast lookup,
- robustness to small perturbations,
- efficient conditional computation.

The loss is "controlled" because the information discarded is often less important than the coarse geometric location of the point.

Many high-dimensional learning systems appear to benefit from this kind of coarse geometric abstraction.

---

# 5. Relationship between the binary code and the original projection vector

Suppose we compute many projections

$s=(s_1, s_2, \dots , s_k)$

and binarize:

$b_i = \text{sign}(s_i)$

The binary vector can be viewed as a quantized approximation to the original projection vector.

Replacing each projection by only its sign maps the point onto a corner of a k-dimensional hypercube.

Although magnitude information is lost, angular information is partially preserved.

A famous result for random hyperplane hashing states:

> The probability that two vectors receive different bits equals their angle divided by π.

Specifically,

$P(b_x \neq b_y) = \theta / \pi$

where θ is the angle between vectors.

Thus the Hamming distance between binary codes estimates angular distance.

This is one reason random hyperplane LSH works so well.

The binary representation is therefore not arbitrary compression. It is a geometric approximation preserving neighbourhood structure.

---

## 6. Atlas networks: LSH for conditional parameter selection

In a conventional dense layer, every example uses essentially the same parameter matrix.

In an Atlas LSH network, the LSH bits can determine:

- which weights participate,
- which neurons activate,
- which matrix blocks are used,
- which experts are selected,
- which memory entries are accessed,
- which routing paths are followed.

The hash code acts as a routing signal.

Different regions of input space can therefore activate different subsets of parameters.

Conceptually:

Input geometry → LSH bits → Parameter selection

or

Input geometry → Routing decisions

The network becomes conditionally sparse.

Only a fraction of the total parameter set may be active for any given input.

---

## 7. Random hyperplanes create a hierarchical partition of space

A single hyperplane divides space in two.

Two hyperplanes divide it into several regions.

Many hyperplanes create a very large collection of cells.

With k bits there are potentially

$2^k$

distinct regions.

Deep Atlas networks repeat this process layer after layer.

Each layer may refine the partitioning.

This creates a hierarchical decomposition of the input manifold.

Nearby inputs often follow similar paths through the network.

Distant inputs may diverge rapidly.

---

# 8. Random projection trees

Random projection trees are data structures built using repeated random hyperplane splits.

At each node:

1. choose a random direction,
2. project the data,
3. split according to projection sign or threshold,
4. recurse.

The resulting tree recursively partitions space.

These structures are used for:

- nearest neighbour search,
- clustering,
- approximate retrieval,
- manifold exploration.

Atlas networks can be viewed as learning and exploiting similar geometric partitions, except the partitions directly control computation.

Deep LSH routing resembles traversing an ensemble of random projection trees.

---

# 9. Why random projections work in high dimensions

High-dimensional geometry has surprising properties.

Random directions are often remarkably informative.

Results related to the Johnson–Lindenstrauss lemma show that random projections approximately preserve distances and neighbourhood relationships.

Consequently, many random projections can capture much of the essential geometry of the data without requiring carefully engineered directions.

This is one reason random features, random projections, and LSH have become so influential.

---

# 10. Fast high-dimensional random projections

A dense random projection matrix costs

$$O(nd)$$

operations per projection.

For very large dimensions this can be expensive.

A popular acceleration is

$$HDx$$

where

- H is the Fast Walsh-Hadamard Transform,
- D is a diagonal matrix with random entries ±1.

The random sign matrix D randomly flips coordinates.

The Hadamard transform then mixes all coordinates globally.

Because the Fast Walsh-Hadamard Transform requires only

O(nlog(n))

operations, extremely high-dimensional random projections become practical.

This gives:

- very fast hashing,
- excellent hardware efficiency,
- low memory cost,
- scalability to large Atlas models.

The transform effectively produces pseudo-random global views of the input while avoiding the cost of storing enormous dense random matrices.

---

# 11. Intuition in one sentence

An Atlas LSH neural network can be viewed as a system that repeatedly asks:

> "From many random geometric viewpoints, where in high-dimensional space does this input appear to lie?"

The resulting binary answers determine how information flows and which parameters participate in computation.